정적 테이블 기반의 XML 문서 저장 시스템 개선

권 훈°, 김정희, 곽호영
제주대학교 컴퓨터 공학과
{dreamerz°, carina, kwak}@cheju.ac.kr

The Improvement of XML document repository based on The Static Table

Hoon-Kwon°, Jeong-Hee Kim, Ho-Young Kwak
Dept. of Computer Engineering, Cheju National University

요 약

본 논문에서는 XML 문서를 관계형 데이터베이스에 저장하는 XML 문서 저장 시스템을 제안한다. 제안 시스템은 XML 문서 구조인 DTD(Document Type Definition)를 보완한 XML Schema를 XML 문서의 기반 구조로 사용하며, 또한 XML 문서의 저장과 색인의 효율성 및 유효성 검사를 위해 XML 문서에 대한 저장 구조를 XML Schema와 XML Instance 문서간의 통합 정적 테이블과 필드들을 가지는 형태로 생성, 처리토록 하였으며, 그 결과 XML 문서와 XML Schema간의 구조상의 유효성을 증대 시킬 수 있음을 알 수 있었다.

1. 서 론

최근 인터넷 사용과 정보의 양이 급증하면서 인터넷상의 정보를 보다 효과적으로 사용하고자 하는 연구가 활발히 진행되고 있다. 한편 웹에서 가장 많이 사용되고 있는 HTML은 문서의 구조보다는 표현에 중점을 두고 있어 특정 응용분야의 구조를 표현하는 문서로는 기능이 부족하다는 단점이 있다[3]. 이에 W3C(World Wide Web Consortium)에서는 차세대 웹 문서의 표준으로 XML(Extensible Markup Language)이라는 전자문서 메타 언어를 1996년 제안하였으며, 현재까지 그 기능이 계속 확장되고 있는 상태이다[1].

XML 문서들의 구성요소를 제한하는 기술로는 DTD (Document Type Definitions)와 XML Schame 두 가지가 있다. DTD는 초기 XML 권고안에 포함된 기술로 선언 규칙을 통해 문서의 구조를 정의하는 방식이다. 반면 XML Schema는 요소 템플릿을 이용하여 문서의 구조를 정의하는 방식으로 XML Schema자체도 XML 문법을 사용하여 DTD의 단점들을 보완하여 만들어졌으며 객체지향 개념을 도입하여 보다 강력하게 문서의 구조를 정의할 수 있게 되었다.[1,2,4]

본 논문에서는 효율적인 XML 문서의 저장,색인 및 유효성을 위해 기존 연구의 기초인 DTD 문서 구조를 보완한 XML Schema기반의 XML 인스턴스들을 관계형 데이터베이스에 저장하는 방법을 제안한다. 또한 저장,색인의 효율성을 고려하여 관계형 데이터베이스내의 테이블 구조는 XML 인스턴스를 저장하기 위한 인스턴스와는 별개로 일반적인 XML 기본 구성 요소에 따라 기 생성되도록 하였다. 이는 기존 DTD를 적용한 저장 시스템은 검색을 위한 문서의 구조를 중점으로 한 저장 구조를 가지고 있기 때문에 문서를 관리하는 측면에는 매우 취약하기 때문이다.

본 논문의 구성은 먼저, 2장에서는 제안한 시스템에서 사용할 XML Schema와 XML 인스턴스를 통합 저장할 수 있는 기본적인 저장 구조를 제시하고, 3장에서는 제시한 저장 구조를 적용한 저장 시스템을 기술한다. 4장에서는 기존 DTD 및 XML Schema를 적용한 시스템과 제안 시스템과의 비교분석 하며 마지막으로 5장에서 결론과 향후 연구를 기술한다.

2. XML Schema와 XML 인스턴스의 저장 구조

XML Schema와 XML 인스턴스는 XML의 기본 문법이 적용되므로 기본적인 구조는 같다. 여기서 본 논문의 제안구조는 SDICHK, ID, SID, 계층정보, Element, Attribute, Entity, Namespace, Eltype, Rep를 구성요소로 하였다.

문서 저장구조는 그림1에 나타내었으며, SDICHK는 Schema, DTD, Instance 문서를 나타내는 식별자이며, ID는 엘리먼트들을 구분하기 위한 식별자이며, SID는 XML Schema를 저장할 때에는 스키마간 구분을 위한 구분자로 XML 인스턴스를 저장할 때에는 적용되는 스키마를 구분하기 위해 사용되며 계층정보는 상위 엘리먼트를 가르키는 ref, 왼쪽 형제 엘리먼트를 가르키는 ref_Left, 그리고 엘리먼트의 오른쪽 형제 엘리먼트를 가르키는 ref_Right로 구성되며 Element는 엘리먼트명, Attribute는 엘리먼트의 속성과 그에 해당하는 값들을 가지며, Entity는 엘리먼트가 가지는 값을, 그리고 네임스페이스는 접두사를 갖는 nsName과 URI를 저장하는 URI, 또한, ELTYPE는 엘리먼트의 Type을 명시하기위한 식별자로 구성되도록 하였으며, 마지막으로 Rep는 엘리먼트의 출현빈도를 기억하게 된다.

그림 1. 제안 XML 저장 구조

그림1의 저장 구조를 이용하면 XML Schema정보와 XML 인스턴스를 동시에 그리고 통합 저장하는 것이 가능하다. 위의 구조를 적용하면, 스키마와 스키마에 따른 인스턴스간의 엘리먼트 계층구조 및 ID가 같게 되어, 문서간 유효성여부를 확인할 수 있다. 또한, SID 필드를

두어, XML Schema에서는 해당 인스턴스문서를, 인스턴스 문서에서는 해당 XML Schema를 각각 매칭시킬 수 있도록 하였다.

제안한 구조를 모델링하기 위한 예제 XML Schema는 그림 2와 같으며, 그에 따른 인스턴스 문서는 그림 3과 같다. [5]

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Order">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="OrderName" type="xs:string" />
        <xs:element name="OrderSu" type="xs:string" />
      </xs:sequence>
    </xs:compleType>
  </xs:element>
</xs:schema>
```
①

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Customer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="FirstName" type="xs:string" />
        <xs:element name="Middle" type="xs:string" />
        <xs:element name="LastName" type="xs:string" />
      </xs:sequence>
      <xs:attribute name="customerID" type="integer" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```
②

그림 2. 예제 XML Schema 문서들

```
<?xml version="1.0" ?>
<Order>
  <OrderName>DB book</OrderName>
  <OrderSu>3</OrderSu>
</Customer>
```
①

```
<?xml version="1.0" ?>
<Customer customerID="24332">
  <FirstName>Ray</FirstName>
  <MiddleInitial>G</MiddleInitial>
  <LastName>Bay</LastName>
</Customer>
```
②

그림 3. 예제 XML Instance 문서들

그 두 개의 문서의 구조를 본 논문에서 제시한 저장 구조에 저장을 하게되면, 표 1과 같은 형태로 저장이 된다.

[illegible]	Instance : I , Schema : S , DTD : D
Ellype	CS : Complex Sequence CC : Complex Choice
Rep	출현빈도수를 나타냄 횟수 지정일 경우 : 해당 횟수 1번이상 여러번일 경우 : *

3. XML 문서 저장 시스템 구조

본 논문에서 제안하는 시스템은 그림 4와 같이 크게 Interface Class, Module Class, Analyzer Class, Storage Class 4부분으로 구성된다.

3.1 Interface Class

Interface Class는 외부 요청에 따른 데이터를 입력받고 그에 따른 결과를 보여주는 부분으로서 본 논문에서 제안한 시스템에서는 크게 Search, Input, Modify Interface로 구분하였다.

Search Interface는 저장된 XML 인스턴스 내에서 사용자가 필요로 하는 데이터를 검색 및 출력하기 위한 Interface로서 검색에 필요한 질의 정보를 사용자로부터 입력받아 Search Module로 전달하며, 전달된 정보에 따른 반환 결과를 사용자에게 보여준다.

Input Interface는 저장할 XML 인스턴스와 XML Schema를 사용자로부터 입력 받아서 관계형 데이터베이스에 저장하기 위한 Interface이다.

Modify Interface는 저장된 XML 인스턴스 또는 XML Schema를 수정 및 삭제하기 위한 정보를 사용자로부터 입력받아 Modify Module로 전달하여 명령의 수행 성공 여부를 사용자에게 보여준다.

표 1 저장 구조에 따른 저장내용

S	0/1	0/0/1	xml	version="1.0"				
S	0/1	0/1/0	schema			xs / http://....		
S	1/1	0/0/0	element	name="Order"		xs / http://....	CS	*
S	2/1	1/0/1	element	name="OrderName" type="xs:string"		xs / http://....		
				... ...				
I	0/1	0/0/0	xml	version="1.0"				
I	1/1	0/0/0	Order					1
I	2/1	1/0/1	OrderName		DB book			1
				... ...				
S	0/2	0/0/1	xml	version="1.0"				
S	0/2	0/1/0	schema			xs / http://....		
S	1/2	0/0/0	element	name="Customer"		xs / http://....	CS	*
S	2/2	1/0/1	element	name="FirstName" type="xs:string"		xs / http://....		
				... ...				
S	1A/2	0/0/0	attribute	name="customerID" type="xs:integer"		xs / http://....		
I	0/2	0/0/0	xml	version="1.0"				
I	1/2	0/0/0	Customer	CustomerID="24332"				1
I	2/2	1/0/1	FirstName		Ray			1
				... ...				

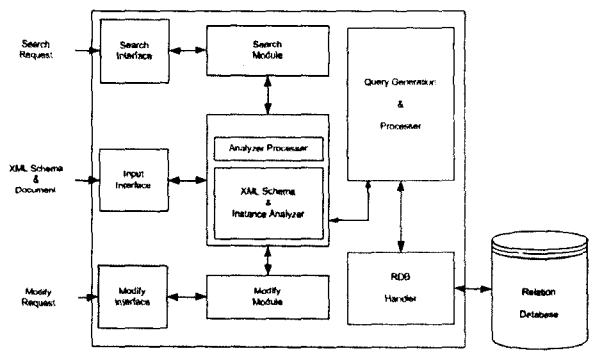

그림 4. 제안 시스템 구성도

3.2 Analyzer Class

Analyzer Class는 Input Interface Class나 Module Class에서 들어온 요청에 따라 이를 분석하여 Query Generation & Processor에서 처리할 수 있는 정보를 생성하는 역할을 하게 되며 Analyzer Processor, XML Schema & XML Analyzer로 구성된다.

작업 요청이 들어오면 Analyzer Processor에서는 XML Schema & XML Analyzer를 이용하여 작업을 수행하기

위한 테이블 정보를 생성하며 이를 Query Generation & Processor에 전달하여 작업을 수행한다.

XML Schema & XML Analyzer는 요청한 작업에 해당하는 XML Schema와 XML 인스턴스에서 작업에 필요한 정보를 얻어오는 역할을 한다.

3.3 Module Class

Module Class는 Search와 Modify Module를 말하며 필요한 연산을 Analyzer부와 연계하여 작업을 수행한다. 검색과 수정에 따른 연산식을 만들어 Analyzer부와 연계하여 작업을 수행하게 된다.

3.4 Storage Class

Storage Class는 Query Generation & Processor과 RDB Handler로 구성되며, 관계형 데이터베이스에 저장하기 위한 Query를 생성하고 이를 처리하는 역할을 한다. Query Generation & Processor부분은 Analyzer부에서 입력받은 정보를 이용해 관계형 데이터베이스를 이용하기 위한 표준 질의를 생성하는 부분이며 RDB Handler 부분은 Query Generation & Processor에서 생성된 Query를 관계형 데이터베이스에 적용하는 부분이다.

3.5. 시스템 처리과정

본 논문에서 제안한 저장 시스템에서 중심이 되는 부분은 Analyzer Class이다. Interface Class나 Module Class에서 요청이 들어오면 Analyzer Class에서는 요청을 분석하여 Query Generation & Processor에 전달한다. Query Generation & Processor에서는 이를 이용해 질의어 생성을 하며 RDB Handler에 질의를 넘겨주며

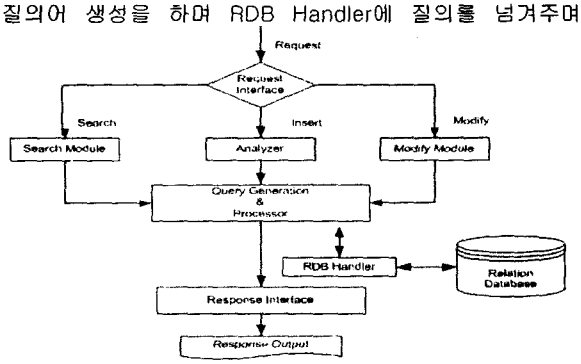

그림 5. 시스템 처리 과정

RDB Handler은 관계형 데이터베이스에 질의를 수행하게 된다. 질의를 수행한 후 결과는 역순으로 사용자에게 전달된다. 처리과정은 그림 5와 같다.

4. 기존 XML 문서 저장 시스템과의 비교

기존 XML 문서 저장 시스템들의 저장방식은 크게 가상분할과 분할저장 방법을 사용하여 DB에 저장을 하였다. 본 제안 시스템은 가상분할의 CLOB 형식으로 저장함과 동시에 각 엘리먼트구조들을 분할 저장 함으로써, 기존 두가지 방법의 상충적인 형태를 지니고 있다. 또한, 기존 DTD를 적용한 저장 시스템은 검색중심의 설계방식을 적용하였으며 별도의 공간에 색인 구조를 생성해야 한다. 따라서, 색인 구조를 저장하기 위한 추가 저장 공간이 필요하며 새로운 인스턴스를 추가로 저장할 때는 색인 구조를 재생성해야 한다. 또한 저장된 문서나 DTD를 변경하는 경우나 저장된 DTD를 적용한 문서의 유효성 검사를 하는 경우 추가연산에 많은 시간이 소요된다. 본 논문에서 제안한 저장 시스템과 DTD를 적용한 시스템과의 비교는 표 2와 같다.

표 2 시스템 비교

구 분	기존 DTD 적용 시스템	제안 시스템
주저장내용	문서 구조 중심	문서 구조/내용
색인구조여부	존재함	존재하지 않음
문서 저장 목적	검 색	문서 관리
저장구조 복잡도	DTD와 문서의 저장 구조가 다르며 검색 위주의 저장방식이기 때문에 구조 정보의 저장비율이 높다.	동일 문법을 사용하는 XML Schema를 적용하여 레벨구조가 문서와 Schema 같게 저장가능하며 저장 시 문서의 손상을 최소화한다.
유효성 검사	문서의 구조와 간단한 값 검사만이 가능함	문서의 구조는 물론 값의 형태와 범위 등 강력한 검사가 가능함.

5. 결 론

본 논문에서는 DTD의 단점을 보완한 XML Schema를 적용하여 XML 인스턴스를 관계형 데이터베이스에 저장할 수 있는 구조와 이를 이용한 저장 시스템을 설계하였다.

제안한 저장 시스템은 정적 테이블의 사용으로 저장 공간 활용이 효율적이며 XML 인스턴스나 XML Schema의 손상을 최소한으로 유지하면서, 인스턴스나 Schema의 레벨구조가 동일하게 저장하기 때문에, 별도의 유효성 검사에 따른 연산시간을 최소화할 수 있다. 또한 정적 테이블의 사용은 저장 구조를 최대한 간단하고 고정된 형태를 유지할 수 있도록 하게 함으로써 이를 이용한 애플리케이션의 개발 시 개발 기간을 단축시킬 수 있을 것이다.

하지만 제안한 저장구조를 적용하면 속성 값의 중복 사용에 따른 유효성체크가 힘들어지며, XML Schema의 모든 기능을 사용하기 어렵다는 단점이 발생하므로 향후 연구는 제안한 구조를 바탕으로 하여 문서자체의 속성값 중복에 따른 유효성 처리 및 속성값 검색에 따른 추가 연구가 필요할 것으로 생각된다.

참 고 문 헌

[1] Extensible Markup Language (XML) 1.0 (Second Edition), "http://www.w3.org/TR/REC-xml"

[2] XML Schema, "http://www.w3.org/XML/Schema"

[3] D. W. Shin, .., "BUS:An Effective Indexing and Retrieval Schema in Structured Documents", in Proc. Digitial Libraries, 1998

[4] Toung Dao, "An Indexing Model for Structured Documents to Support Querues on Content, Structured and Attributes", Proceedings of ADL '98, pp.88-97, 1998.

[5] Jon Duckett외, "XML Schemas", 정보문화사, 2002.